УДК 519.712.1

МЕТОД ПОСТРОЕНИЯ ИЕРАРХИЧЕСКОЙ ПРЕДИКАТНОЙ МОДЕЛИ ПРОЦЕССА

Рудометкина М.Н., Спицын В.Г., Болотова Ю.А.

ФГАОУ ВПО «Национальный исследовательский Томский политехнический университет», Томск, e-mail: mn.rud@inbox.ru.

В статье разрабатывается метод построения предикатной модели гибкого процесса. Общий недостаток рассмотренных методов интеллектуального анализа процессов состоит в том, что они базируются на предположении о «плоскостном» характере процесса. Процесс представляется графом, отражающим последовательность (алгоритм) действий, приводящих к требуемому результату. В данной области прослеживается аналогия с ранними стадиями развития технологий программирования, когда основное внимание уделялось алгоритмизации. Известные традиционные методы интеллектуального анализа процессов требуют доработки применительно к задаче построения модели гибкого процесса. Усовершенствованный метод должен обеспечивать учет дополнительного аспекта процесса – иерархии его действий, а также возможность эффективного отсечения избыточных действий при конфигурировании модели процесса. В качестве математической базы используется аппарат конечных предикатов.

Ключевые слова: предикат, алгебра предикатов, граф, гибкий процесс, лог процесса, process mining

METHOD FOR CONSTRUCTING A HIERARCHICAL PREDICATE PROCESSING MODEL

Rudometkina M.N., Spitsyn V.G., Bolotova Y.A.

National Research Tomsk Polytechnic University, Tomsk, e-mail: mn.rud@inbox.ru

We develop a method of constructing the predicate model flexible process. The general lack of all considered methods is they are based on the assumption of «planar» nature of the process. The process is represented by a graph reflecting the sequence of actions (an algorithm), that leads to the desired result. In this area, the analogy can be traced to the early stages of programming technologies, when attention was focused on algorithmization. Traditional methods of mining processes need to be improved in relation to the problem of constructing a model of a flexible process. An improved method should take into account an additional aspect of the process – the hierarchy of his actions, and the ability to effectively cut off the excess actions when configuring the process model. As a mathematical framework used machine finite predicates.

Keywords: predicate, logical network, predicate algebra, graph, flexible process, the log process, process mining

В данной работе рассматривается задача построения методами process mining моделей гибких дискретных процессов преобразования ресурсов, структура которых может изменяться как на этапе конфигурирования, так и во время выполнения.

Актуальность данной задачи объясняется следующими положениями:

1) появление моделей гибких процессов обеспечивает дополнительные возможности;

2) иерархическое представление модели гибкого процесса облегчает понимание модели, позволяет использовать в качестве элементов модели существующие формализованные подпроцессы;

3) построение моделей реально выполняющихся процессов обработки ресурсов на основе логов осуществляется методами интеллектуального анализа процессов;

4) существующие методы process mining основаны на использовании аппарата сетей Петри и ориентированы преимущественно на построение моделей дискретных процессов с жестко заданной структурой;

5) перспективным в рамках искусственного интеллекта представляется развитие аппарата алгебры конечных предикатов как средства для идентификации структур для решения задачи построения моделей гибких процессов.

Анализ базовых исследований в данной области

Выполненный анализ базовых исследований в области построения моделей гибких процессов [2, 5, 6, 7, 8] показал, что полная модель такого процесса может быть получена с помощью традиционного полного цикла разработки модели гибкого многовариантного процесса; слияния моделей нескольких «жестких» процессов, которые реализуют идентичную функциональность, но различным способом. Модель строится на основе анализа логов, содержащих информацию о нескольких вариантах реализации процесса [9]. В данной области прослеживается аналогия с ранними стадиями развития технологий программирования, когда основное внимание уделялось алгоритмизации. В дальнейшем произошел переход к объектно-ориентированным технологиям разработки [1].

Предикатная модель гибкого процесса

Использование иерархической модели обеспечивает однозначное определение ситуаций выбора, а также позволяет избежать тупиков. Возможность выбора и отсечения различных вариантов процесса также определяется рассмотренной ранее иерархической структурой предикатов.

Предлагаемый автором метод основан на алгоритме построения блок-структурированного процесса и использует свойство рекурсивности процессного дерева. Как было показано в [3], с каждым узлом данного дерева связано либо конкретное действие процесса, либо один из базовых операторов, задающих взаимосвязи между действиями процесса. Указанные операторы реализованы в виде предикатов АКП.

Действия процесса в предлагаемой модели могут быть описаны как переменной, так и предикатом. Представление действия в виде переменной выполняется для простых действий, которые могут описываться конечным количеством состояний:

$$R_k(x_i, x_j) \in M, \quad k = \overline{1,m},$$

$$x_i, x_j \in A, \quad A = \{a_1, a_2, a_3, a_4, a_5\}, \quad (1)$$

где a_1 = «ожидание»; a_2 = «готовность»; a_3 = «выполнение»; a_4 = «приостановка»; a_5 = «завершено».

В том случае, если действие процесса является составным и состоит из отдельных элементарных операций, то такое действие представляется в виде соответствующей дополнительной составной переменной предиката. Переменными для такого предиката выступают состояния операций, входящих в состав действия процесса. Например, введем две составные переменные $x' = x_1 \wedge x_2$ и $x'' = (x_3 \vee x_4) \wedge x_5$. Тогда последовательность предикатов

$$R_1(x_1, x_2) \to R_2(x_2, x_3) \to R_3(x_3, x_4) \to R_5(x_4, x_5)$$

можно заменить на составной предикат

$$R'(x', x''). \quad (2)$$

Взаимосвязь действий выполняется с помощью предикатов, реализующих базовые структурные элементы процессной модели, как было показано в [3, 4].

Пусть $M_i, i = \overline{1,N}$ – N моделей дерева процессов, каждая из которых представлена в виде иерархии предикатов на конечном множестве действий i-го процесса и конечном множестве базовых структурных элементов:

$$M_i = \{R_k^i \mid \forall R_k(x_i, x_j) \in M_i, x_i \in V^*, i, j = \overline{1,n}, \ \exists P_k \in P, k = \overline{1,m}\}, \quad (3)$$

Определим предикат M по правилам АКП, аргументами которого являются модели M_i.

$$M(M_1, M_2, ..., M_N) = M_1 \wedge M_2 \wedge ... \wedge M_N. \quad (4)$$

Тогда предикат (4) также является деревом процессов, а входящие в его состав модели можно рассматривать как подпроцессы текущего процесса.

Истинность входящих в состав данного предиката моделей M_i означает выполнимость (достижение его конечного состояния) процесса при допустимых исходных данных.

Следовательно, предикат M будет истинным в том случае, если истинными будут все его аргументы – модели процессов M_i, что и показывает выполнимость интегрального процесса M.

В свою очередь предикат (4) также может быть аргументом у предиката более высокого уровня, что и позволяет рекурсивно определить предикатную модель процесса.

Очевидно, что слияние фрагментов предикатной модели осуществляется путем использования предиката более высокого уровня, аргументами которого являются предикаты, формализующие сливаемые фрагменты:

$$K(K_1, K_2, ..., K_S) = K_1 \wedge K_2 \wedge ... \wedge K_S. \quad (5)$$

Критерием адекватности полученной модели гибкого процесса является способность воспроизводить все траектории исходных логов, которые были использованы в качестве входных данных при построении модели:

$$\forall R'(x', x'') \in M \ \ \exists K_j \subset ... \subset M_i \subset M \ R'(x', x'') \in K_j. \quad (6)$$

Будем рассматривать две предикатные модели гибкого процесса в качестве изоморфных в том случае, если из обеих могут быть получены идентичные логические сети:

$$M' \sim M'', \ M' = \langle S'_k, R'_i \rangle, \ M'' = \langle S''_k, R''_i \rangle, \quad (7)$$

если $(S'_k \sim S''_k) \wedge (R'_i \sim R''_i)$.

Метод построения модели гибкого процесса требует предопределенных исходных данных и включает в себя ряд шагов.

В качестве исходных данных выступает лог гибкого процесса L, полученный слиянием k традиционных логов.

Этап 1. При анализе полного лога L выполняется поиск возможностей его разбиения на более мелкие фрагменты L_i, которые в дальнейшем могут быть объединены последовательно, параллельно, циклически или через выбор согласно заданным предикатным операциям [4]:

$$\begin{aligned} S_1 &= R_1(x_1, x_2) \oplus R_2(x_1, x_3); \\ S_2 &= R_3(x_2, x_4) \vee R_4(x_2, x_5); \\ S_3 &= R_7(x_5, x_7) \vee R_8(x_5, x_8); \\ S_4 &= R_{10}(x_7, x_9) \wedge R_{11}(x_7, x_{10}); \\ S_5 &= \Omega(R_{15}(x_{10}, x_{12}), R_{16}(x_{10}, x_{14}), 3). \end{aligned} \quad (8)$$

Основные, достаточно очевидные ограничения, которые учитываются при разбиении лога на составляющие:

1. Объединение (обратная к разбиению операция) полученных в результате разбиения логов составляет исходный лог либо является надмножеством для исходного лога L;

$$L \subseteq (L_1 \cup L_2 \cup \ldots \cup L_m). \quad (9)$$

2. Длина *Length* любого полученного фрагмента L_i должна быть не больше длины исходного лога:

$$Length(L) \geq Length(L_i)\ \forall i = \overline{1, m}. \quad (10)$$

Общий подход к разбиению заключается в выделении последовательностей событий, либо входит в состав более сложных конструкций – выбор, параллельное выполнение, цикл:

$$\begin{gathered} \text{если } \exists S(R_i, R_j) \Rightarrow \exists L_1, L_2, \text{ такие, что } R_i \in L_1 \text{ и } R_i \in L_2 \text{ и} \\ R_i - \text{начало фрагментов } L_1, L_2. \end{gathered} \quad (11)$$

Этап 2. Построение предикатной модели на текущем уровне, когда аргументами предиката являются выделенные на этапе 1 фрагменты. Сами выделенные фрагменты еще не формализованы:

$$\begin{gathered} \text{если } \exists L_1, L_2, \text{ такие, что } R_i \in L_1 \text{ и } R_i \in L_2 \text{ и} \\ \exists S(R_i, R_j), \text{ то } L = L_1 \cup L_2. \end{gathered} \quad (12)$$

Этап 3. Проверка текущего фрагмента лога: может ли он быть разделен на более мелкие фрагменты:

$$\exists(L_1, L_2, \ldots, L_m), \quad (13)$$

что $\langle (L_1, L_2, \ldots, L_m), (S_1, S_2, \ldots, S_5) \rangle \Leftrightarrow M$.

Этап 4. Если фрагмент лога разделен быть не может, то выбор следующей составляющей модели согласно выражению (10), после чего выполняется переход к этапу 1.

Этап 5. Проверка: все ли доступные фрагменты разделены на составляющие. Если результат положительный, то модель процесса в виде предикатного дерева построена.

Первый базовый элемент отражает отношение непосредственного следования действий процесса. Поэтому работа алгоритма начинается с построения предикатных моделей последовательности операций процесса, каждая из которых объединяет только такие последовательности операций процесса, для которых в логе содержится соответствующая последовательность событий.

$$R_1(x_1, x_2) \wedge R_2(x_2, x_3) \wedge \ldots \wedge R_n(x_n, x_{n+1}). \quad (14)$$

Рассмотрим лог, который представлен в виде следующего набора последовательностей событий («следов» выполнения процесса):

$$L = \{< s_1, s_2, s_3 >, < s_1, s_3, s_2 >, < s_1, s_4, s_5, s_6, s_4, s_5 >, < s_1, s_7, s_8 >\}. \quad (15)$$

Данный лог отражает следующие возможные элементы процесса:

– ветвление начиная от события s_1, поскольку за данным событием в каждом из следов возникают события s_2 либо s_3, либо s_4, либо s_7;

– параллельное выполнение действий, которые отражены событиями s_2 и s_3, поскольку в первой строке событие s_2 предшествует событию s_3, а во второй – наоборот;

– цикл, поскольку в последней строке пара событий s_4, s_3 повторяется после события s_6;

– последовательности событий s_7, s_8.

Модель процесса, позволяющая получить рассмотренный выше лог, представляется в виде такого набора предикатов:

$$\begin{cases} R_1(x_1, x_2) \wedge R_2(x_2, x_3); \\ R_3(x_1, x_3) \wedge R_4(x_3, x_2); \\ R_5(x_1, x_4) \wedge R_6(x_4, x_5) \wedge R_7(x_5, x_6) \wedge R_8(x_6, x_4) \wedge R_9(x_4, x_5); \\ R_{10}(x_1, x_7) \wedge R_{12}(x_7, x_8); \\ S_1^{OR}(R_1, R_2, R_3, R_4, R_7); \\ S_2^{AND}(R_2, R_3); \\ S_3^{\Omega}(R_4, R_5). \end{cases} \quad (16)$$

Для нахождения указанной модели целесообразно отдельно выделить ветвление, которое во всех случаях в примере начинается с события s_1, а также последовательности событий $\langle s_2, s_3\rangle$, $\langle s_3, s_2\rangle$, $\langle s_4, s_5, s_3, s_6, s_4, s_5\rangle$, $\langle s_7, s_8\rangle$, отражающие параллельное выполнение, простую последовательность и цикл соответственно. Рассматривая каждую из этих групп как единое целое, сворачиваем наш лог к набору простых последовательностей событий, каждое из которых отражается в модели в виде предиката:

$$\begin{aligned} &R_1'(x_1, x_3) = R_1(x_1, x_2) \wedge R_2(x_2, x_3); \\ &R_2'(x_1, x_2) = R_3(x_1, x_3) \wedge R_4(x_3, x_2); \\ &R_3' = R_5(x_1, x_4) \wedge R_6(x_4, x_5) \wedge R_7(x_5, x_6) \wedge R_8(x_6, x_4) \wedge R_9(x_4, x_5); \\ &R_4' = R_{10}(x_1, x_7) \wedge R_{12}(x_7, x_8). \end{aligned} \quad (17)$$

Далее определяется модель для каждой из последовательностей $\langle s_2, s_3\rangle$, $\langle s_3, s_2\rangle$, $\langle s_4, s_5\rangle$, $\langle s_4, s_5, s_3, s_6, s_4, s_5\rangle$, $\langle s_7, s_8\rangle$. При этом для параллельного выполнения мы разделяем события s_2 и s_3, для цикла – выделяем отдельно повторяющиеся последовательности $\langle s_4, s_5\rangle$, а также событие s_6, которое предположительно соответствует проверке условия повторения.

Обобщив рассмотренный пример, получаем, что для выделения типовых фрагментов необходимо выполнить следующие основные шаги:

1) разделить лог на последовательности событий, которые не содержат ветвлений;

2) разделить фрагменты лога на последовательности событий, которые не содержат циклов и параллельного выполнения;

3) выделить базовые элементы ветвления, цикла, параллельности между полученными на втором шаге последовательностями событий и представить их в виде предикатов.

Для реализации третьего шага необходимо формализовать признаки, характеризующие наличие в логе ветвления, цикла, параллельного выполнения.

Предварительно доопределим рассмотренные ранее последовательности событий, указав для каждой начальные и конечные события. Тогда для каждой выделенной последовательности событий Λ_i существуют начальное s_i^* и конечное события s_i^{**} так что

$$\forall s_{i,j} \exists s^*, s^{**} : s_i^* \le s_{i,j}, s_{i,j} \le s_i^{**} \mid s_i^*, s_i^{**} s_{i,j} \in \Lambda_i, \quad (18)$$

где знак $\le$ задает отношение упорядоченности событий во времени (т.е. порядок записи событий в логе).

Теперь определим последовательность, состоящую из произвольного количества событий и отражающую различные версии реализации процесса. Пусть имеем набор упорядоченных последовательностей событий $\Lambda = \{\Lambda_i\}$, $\Lambda_i = \left\langle s_i^*, ..., s_{i,j}, ..., s_i^{**} \right\rangle$. Каждая из указанных последовательностей отражает одну из реализаций процесса.

Тогда последовательность действий для процесса выявляется следующим образом. Во-первых, существует выделенный набор последовательностей событий лога $\Lambda = \{\Lambda_i\}$, имеющих идентичные начальные и конечные события. Во-вторых, порядок событий не нарушается в каждой из последовательностей Λ_i:

$$\forall (s_m, s_n) \in \Lambda_i \neg \exists (s_n, s_m) \in \Lambda_j, \mid \left(s^{**}, s^*\right) \in \Lambda_i, \Lambda_j, \ \Lambda_i, \Lambda_j \in \Lambda \quad (19)$$

Параллельное выполнение действий процесса является антиподом для последовательности и потому выявляется в том случае, если порядок для пар событий в различных последовательностях из множества Λ отличается:

$$\forall(s_m, s_n) \in \Lambda_i \exists(s_n, s_m) \in \Lambda_j, | \left(s^{**}, s^{*}\right) \in \Lambda_i, \Lambda_j, \ \Lambda_i, \Lambda_j \in \Lambda. \quad (20)$$

Выбор (ветвление) определим следующим образом. Пусть имеем набор упорядоченных последовательностей событий $\Lambda = \{\Lambda_i\}$, которые имеют общее начальное событие s^*. Набор Λ отражает ветвление в том и только в том случае, если последовательности Λ_i не имеют общих событий за исключением начального s^*:

$$(\exists s^* \in \Lambda_i, \Lambda_j) \wedge (\forall s_n \in \Lambda_i \quad s_n \notin \Lambda_j), \Lambda_i, \Lambda_j \in \Lambda. \quad (21)$$

Теперь определим цикл. Последовательность событий Λ_i отражает выполнение цикла в модели процесса при выполнении следующих условий:

1) последовательность Λ_i не имеет общих событий с другими последовательностями множества Λ:

$$\forall s_m \in \Lambda_i, s_m \notin \Lambda_j, s_m \neq s^*, s_m \neq s^{**}; \quad (22)$$

2) начальное событие текущей последовательности Λ_i может иметь последовательную связь с событием из другой последовательности Λ_j:

$$\exists s_n \in \Lambda_j : s_n \to s^{**}, s \ \in \Lambda_i, \ \Lambda_i, \Lambda_j \in \Lambda; \quad (23)$$

3) конечное событие текущей последовательности Λ_i может иметь последовательную связь с событием из другой последовательности Λ_j:

$$\exists s_n \in \Lambda_j : s^* \to s_n, s^* \in \Lambda_i, \ \Lambda_i, \Lambda_j \in \Lambda. \quad (24)$$

4) существует повторяющаяся не менее 2-х раз последовательность событий, являющаяся подмножеством исходной последовательности Λ_i:

$$\begin{gathered} \exists\{s_n, s_{n+1}, ..., s_{n+i}\} \in \Lambda_j; \\ \exists(s_n \to s_{n+1} \to ... \to s_{n+i}); \\ \exists(s_{n+i} \to s_n). \end{gathered} \quad (25)$$

Приведенное в начале параграфа ограничение на полноту лога после описания метода можно задать следующим образом: каждое действие каждого варианта процесса должно быть представлено в логе $L = \{l_k\}$ (т.е. в каждом варианте) хотя бы один раз:

$$\forall s^* \to s^{**} \in \Lambda_i \Rightarrow s^* \to s^{**} \in \Lambda. \quad (26)$$

Заключение

В статье предложен рекурсивный метод построения иерархической предикатной модели гибкого процесса на основе анализа логов, который при условии полноты лога обеспечивает построение полного адаптируемого описания процесса в форме иерархии предикатов, связывающих его базовые структурные элементы. Метод обеспечивает возможность проверки адекватности модели в процессе ее построения путем решения системы уравнений, в качестве исходных данных которой выступают логи выполнившегося процесса. Область практического применения предлагаемой модели – интеллектуальный анализ процессов.

Исследование выполнено при поддержке РФФИ в рамках научного проекта № 12-08-00296.

Список литературы

1. Аврамчук В.С., Гергет О.М., Лунева Е.Е. Разработка компонента визуализации биомедицинских данных на основе технологии OPENGL // Известия Волгоградского государственного технического университета. – 2013. – Т. 17. – № 14 (117). – С. 28–31.

2. Репин В., Елифиров В. Процессный подход к управлению. Моделирование бизнес-процессов. – М.: Стандарты и качество. – 2008. – С. 196.

3. Рудометкина М.Н. Предикатная модель гибкого процесса // Международный журнал прикладных и фундаментальных исследований. – 2014. – № 10, Ч. 2.

4. Рудометкина М.Н., Спицын В.Г. Модель логической сети с предикатными операциями // Международный журнал прикладных и фундаментальных исследований. – 2014. – № 10, Ч. 2.

5. Gottschalk F.,W.M.P. van der Aalst, and M.H. Jansen-Vullers. Merging Event-driven Process Chains // OTM 2008, Part I, CoopIS 2008, volume 5331 of Lecture Notes in Computer Science, Berlin Heidelberg. – Springer Verlag. – 2008. – P. 418–426.

6. Rosa M.La, Dumas M., Uba R., and Dijkman R. Business Process Model Merging: An Approach to Business Process

Consolidation. ACM Transactions on Software Engineering and Methodology – 2012 – № 22(2).

7. Li C., Reichert M., Wombacher A. The MINADEPT Clustering Approach for Discovering Reference Process Models Out of Process Variants // International Journal of Cooperative Information Systems. – 2010. –№ 19(3–4). – P. 159–203,

8. Gerget O.M., Devjatyh D.V. Information System for Health State Diagnostics // CSIT'2013. – Уфа, 2013. – P. 126–131.

9. Rudometkina M.N., Spitsyn V.G. Detection of processing model basic elements in intellectual analysis of flexible processes // IFOST, The 9th International Forum on Strategic Technology. – 2014.

References

1. Avramchuk V.S., Gerget O.M., Luneva E.E. Razrabotka komponenta vizualizacii biomedicinskih dannyh na osnove tehnologii OPENGL // Izvestija Volgogradskogo gosudarstvennogo tehnicheskogo universiteta. 2013. T. 17. no. 14 (117). pp. 28–31.

2. Repin V., Elifirov V. Processnyj podhod k upravleniju. Modelirovanie biznes-processov //M.: Standarty i kachestvo. 2008. pp. 196.

3. Rudometkina M.N. Predikatnaja model' gibkogo processa. Mezhdunarodnyj zhurnal prikladnyh i fundamental'nyh issledovanij no. 10, chast' 2, Rossijskoj Akademii Estestvoznanija, Moskva, 2014. Prinjata k pechati.

4. Rudometkina M.N., Spicyn V.G. Model' logicheskoj seti s predikatnymi operacijami. Mezhdunarodnyj zhurnal prikladnyh i fundamental'nyh issledovanij no. 10, chast' 2, Rossijskoj Akademii Estestvoznanija, Moskva, 2014. Prinjata k pechati.

5. Gottschalk F.,W.M.P. van der Aalst, and M.H. Jansen-Vullers. Merging Event-driven Process Chains // OTM 2008, Part I, CoopIS 2008, volume 5331 of Lecture Notes in Computer Science, , Berlin Heidelberg. Springer Verlag. 2008. pp. 418–426.

6. Rosa M. La, Dumas M., Uba R., Dijkman R. Business Process Model Merging: An Approach to Business Process Consolidation. ACM Transactions on Software Engineering and Methodology // no. 22(2), 2012.

7. Li C., Reichert M., Wombacher A. The MINADEPT Clustering Approach for Discovering Reference Process Models Out of Process Variants. International Journal of Cooperative Information Systems, no. 19(3-4):159–203, 2010.

8. Gerget O.M., Devjatyh D.V. Information System for Health State Diagnostics // CSIT'2013 2013. pp. 126–131.

9. Rudometkina M.N., Spitsyn V.G. Detection of processing model basic elements in intellectual analysis of flexible processes // IFOST, The 9th International Forum on Strategic Technology 2014.

Рецензенты:

Кочегуров В.А., д.т.н., профессор, ФГАОУ ВПО «Национальный исследовательский Томский политехнический университет» Министерства образования и науки РФ, г. Томск;

Кориков А.М., д.т.н., профессор, заведующий кафедрой АСУ факультета систем управления, ФГБОУ ВПО «Томский государственный университет систем управления и радиоэлектроники», г. Томск.

Работа поступила в редакцию 23.09.2014.